/\* Ass 3: Create BST ,its mirror image ,display it levelwise,display leaf nodes,find height of BST \*/

#include <iostream>

using namespace std;

struct tnode{

int info; // node value

struct tnode \*left; //left child pointer

struct tnode \*right; //right child pointer

};

class bintree

{

tnode \*root;

public:

bintree() //constructor

{

root =NULL;

}

int isEmpty() //function to check if tree is empty

{

if(root==NULL)

return 1;

return 0;

}

void printGivenLevel(tnode\* root, int level);

void printLevelOrder(); //print levelwise

void maxDepth1(); //prints height of tree wrapper

void plnode(); //print leaf nodes wrapper

void printLeafNodes(tnode\* root);

int maxDepth(tnode\* node);

void insert(int val); //function for insertion of node in tree

void mirror1();

void mirror(struct tnode \*);//create mirror image wrapper

};

void bintree::plnode()

{

cout<<endl<<"leaf nodes are:";

printLeafNodes(root);

}

void bintree:: printLeafNodes(tnode \*root)

{

// if node is null, return

if (!root)

return;

// if node is leaf node, print its data

if (!root->left && !root->right)

{

cout << root->info << " ";

return;

}

// if left child exists, check for leaf

// recursively

if (root->left)

printLeafNodes(root->left);

// if right child exists, check for leaf

// recursively

if (root->right)

printLeafNodes(root->right);

}

void bintree::insert(int v)

{

tnode \*ctnode = new tnode;

tnode \*parent;

ctnode->info =v;

ctnode->left=NULL;

ctnode->right=NULL;

parent=NULL;

if(isEmpty())

{

root=ctnode;

}

else

{

tnode \*p=root;

while(p!=NULL)

{

parent =p;

if(v>p->info)

p=p->right;

else

p=p->left;

}

if(v<parent->info)

parent->left=ctnode;

else

parent->right=ctnode;

}

}

void bintree::printGivenLevel(tnode\* root, int level)

{

if (root == NULL)

return;

if (level == 1)

cout<<root->info<<" ";

else if (level > 1) {

printGivenLevel(root->left, level - 1);

printGivenLevel(root->right, level - 1);

}

}

void bintree::printLevelOrder()

{

int h = maxDepth(root);

int i;

cout<<endl<<"level wise"<<endl;

for (i = 1; i <= h; i++) {

printGivenLevel(root, i);

cout<<endl;

}

}

void bintree:: maxDepth1()

{

int d = maxDepth(root);

cout<<"height:"<<d;

}

int bintree:: maxDepth(tnode\* node)

{

if (node == NULL)

return 0;

else {

/\* compute the depth of each subtree \*/

int lDepth = maxDepth(node->left);

int rDepth = maxDepth(node->right);

/\* use the larger one \*/

if (lDepth > rDepth)

return (lDepth + 1);

else

return (rDepth + 1);

}

}

void bintree:: mirror1()

{

cout<<endl<<"mirror image is :";

mirror(root);

printLevelOrder();

}

void bintree:: mirror(struct tnode \*node)

{

if(node==NULL)

{

return;

}

else

{

struct tnode \*temp;

mirror(node->left);

mirror(node->right);

temp=node->left;

node->left=node->right;

node->right=temp;

}

}

int main() {

bintree b;

int ch,ch1,n,d;

do{

cout<<"1.Create Binary Search Tree (BST)"<<endl;

cout<<"2.Insert a node in BST "<<endl;

cout<<"3.Display BST levelwise"<<endl;

cout<<"4.Create Mirror image of BST"<<endl;

cout<<"5.Display leaf nodes"<<endl;

cout<<"6.Display height of BST"<<endl;

cout<<"Enter your choice : ";

cin>>ch1;

switch(ch1)

{

case 1: cout<<"How many nodes in BST : ";

cin>>n;

for(int i=0;i<n;i++)

{

cout<<"Enter data for node";

cin>>d;

b.insert(d);

}

break;

case 2: cout<<"Enter data for node";

cin>>d;

b.insert(d);

break;

case 3:

b.printLevelOrder();

break;

case 4:

b.mirror1();

break;

case 5:

b.plnode();

break;

case 6:

b.maxDepth1();

break;

default:

cout<<endl<<"Enter valid choice";

break;

}

cout<<endl<<"Do you want to continue? Press 1 to continue else 0 :";

cin>>ch;

}while(ch==1);

return 0;

}